(12) UK Patent Application (19) GB (11) 2 229 832 (13) A

(43) Date of A publication 03.10.1990

(21) Application No 9003070.1

(22) Date of filing 12.02.1990

(30) Priority data
(31) 331640 (32) 30.03.1989 (33) US

(71) Applicant
Intel Corporation

(Incorporated in the USA - California)

3065 Bowers Avenue, Santa Clara, California 95051, United States of America

(72) Inventors
John H Crawford
Mustafiz R Choudhury

(74) Agent and/or Address for Service
Potts Kerr and Co
15 Hamilton Square, Birkenhead, Merseyside, L41 6BR, United Kingdom

(51) INT CL[5]
G06F 7/00

(52) UK CL (Edition K)
G4A AFD ASN

(56) Documents cited
GB 1029880 A

(58) Field of search
UK CL (Edition J) G4A ACX AFD AGB ASN
INT CL[4] G06F

(54) Byte swap instruction for memory format conversion within a microprocessor

(57) A microprocessor has an instruction for performing an in-place byte swap on 32-bit data type to convert data stored in a big-endian memory format to a little-endian memory format, or vice-versa. A modified barrel shifter includes a plurality of multiplexers for selectively coupling data from one or more input buses to an output bus. The coupling of the individual bit lines of the data buses is arranged such that the lower order bits of the 32-bit quantity are exchanged with the higher order bits. Control lines connected to each of the multiplexers provide a means for controlling the byte swapping operation.

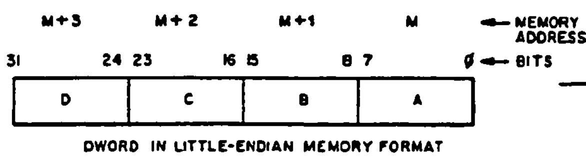


FIG_1

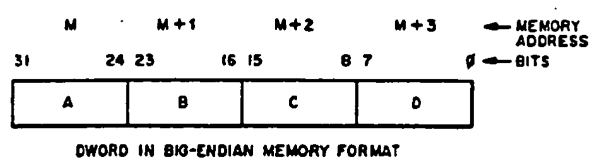
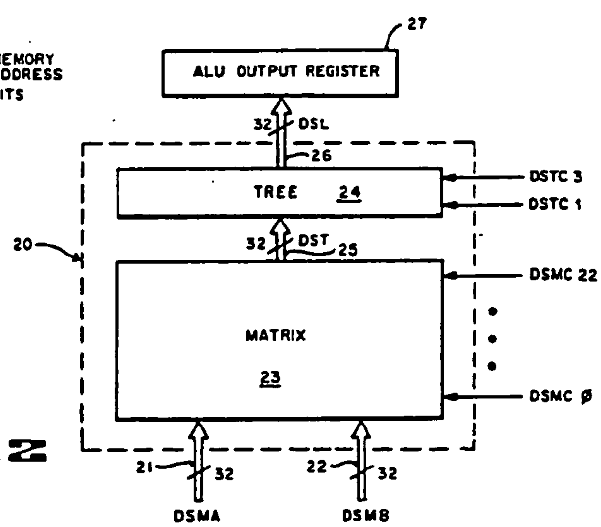


FIG_2



At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

GB 2 229 832 A

FIG_1

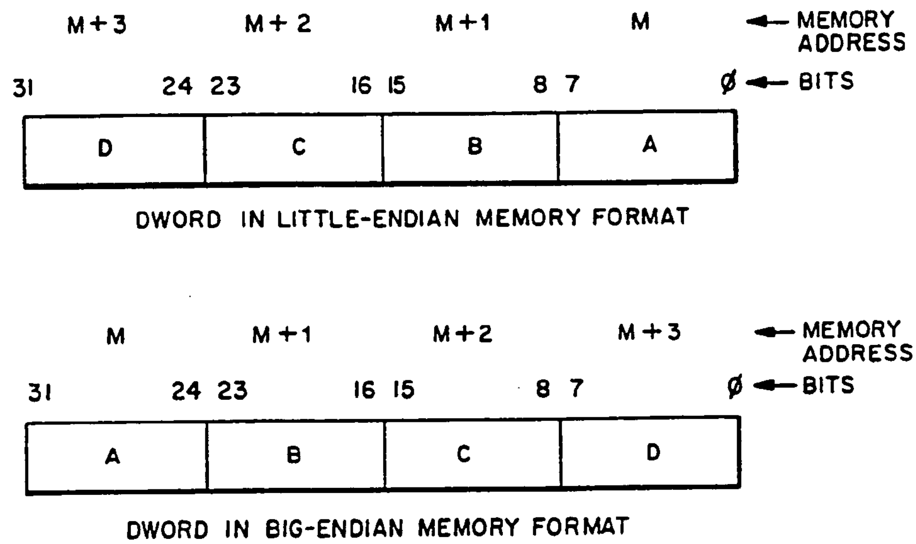

M+3
M+2
M+1
M
MEMORY ADDRESS
31
24 23
16 15
8 7
Ø
BITS
D
C
B
A
DWORD IN LITTLE-ENDIAN MEMORY FORMAT
M
M+1
M+2
M+3
MEMORY ADDRESS
31
24 23
16 15
8 7
Ø
BITS
A
B
C
D
DWORD IN BIG-ENDIAN MEMORY FORMAT

FIG_2

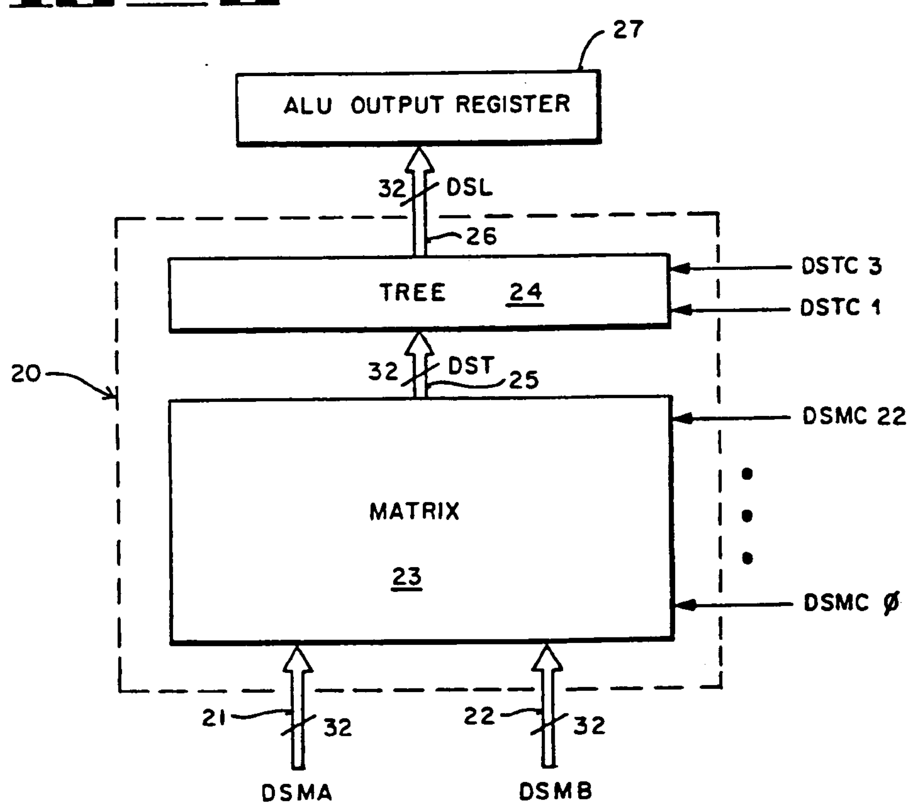

27
ALU OUTPUT REGISTER
32
DSL
26
TREE 24
DSTC 3
DSTC 1
32
DST
25
20
DSMC 22
MATRIX
23
DSMC Ø
21
32
22
32
DSMA
DSMB

FIG 3A

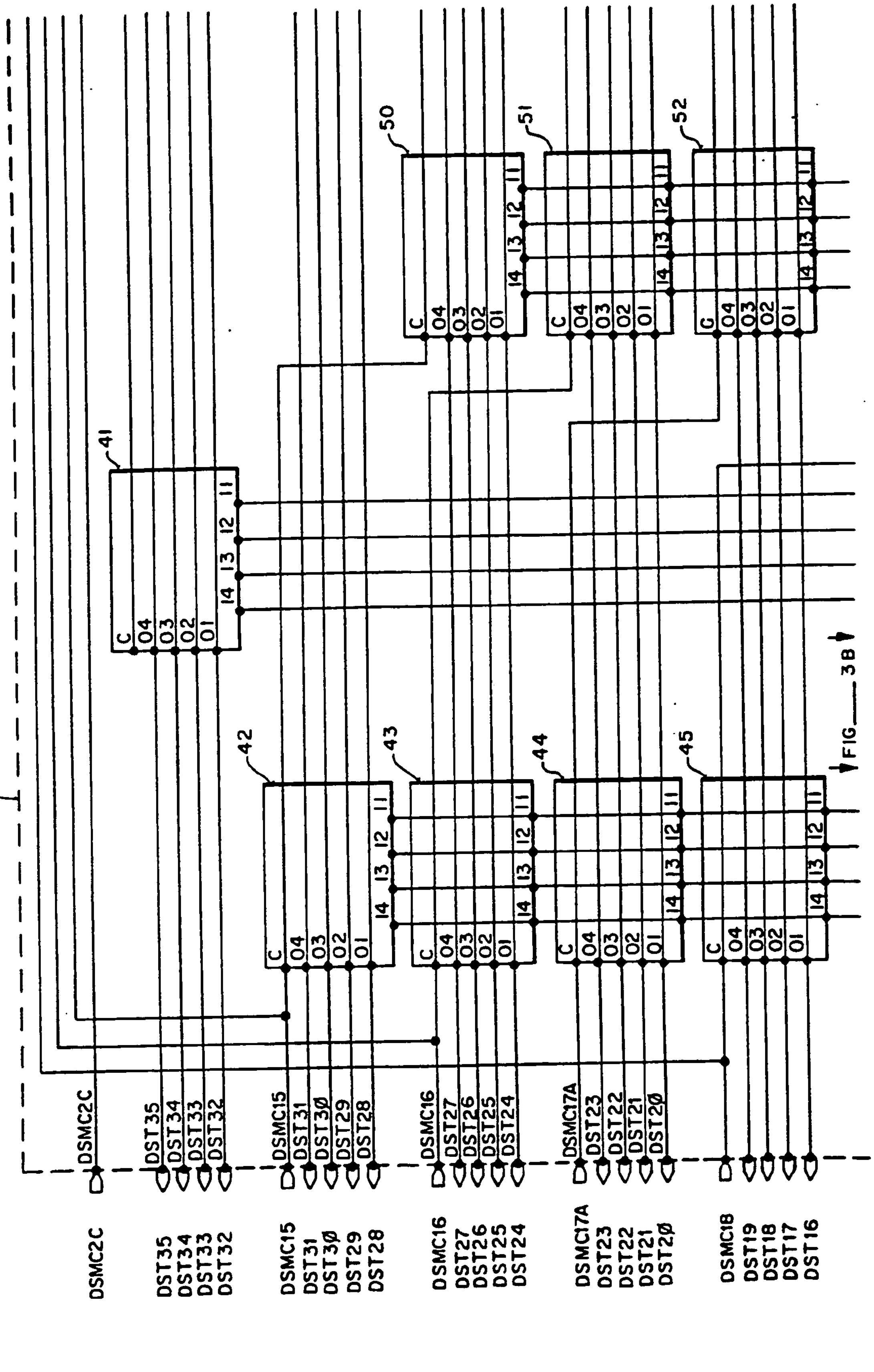


FIG 3C
FIG 3B
20
41
42
43
44
45
50
51
52
C
O4
O3
O2
O1
I1
I2
I3
I4
DSMC2C
DST35
DST34
DST33
DST32
DSMC15
DST31
DST3Ø
DST29
DST28
DSMC16
DST27
DST26
DST25
DST24
DSMC17A
DST23
DST22
DST21
DST2Ø
DSMC18
DST19
DST18
DST17
DST16

FIG 3B

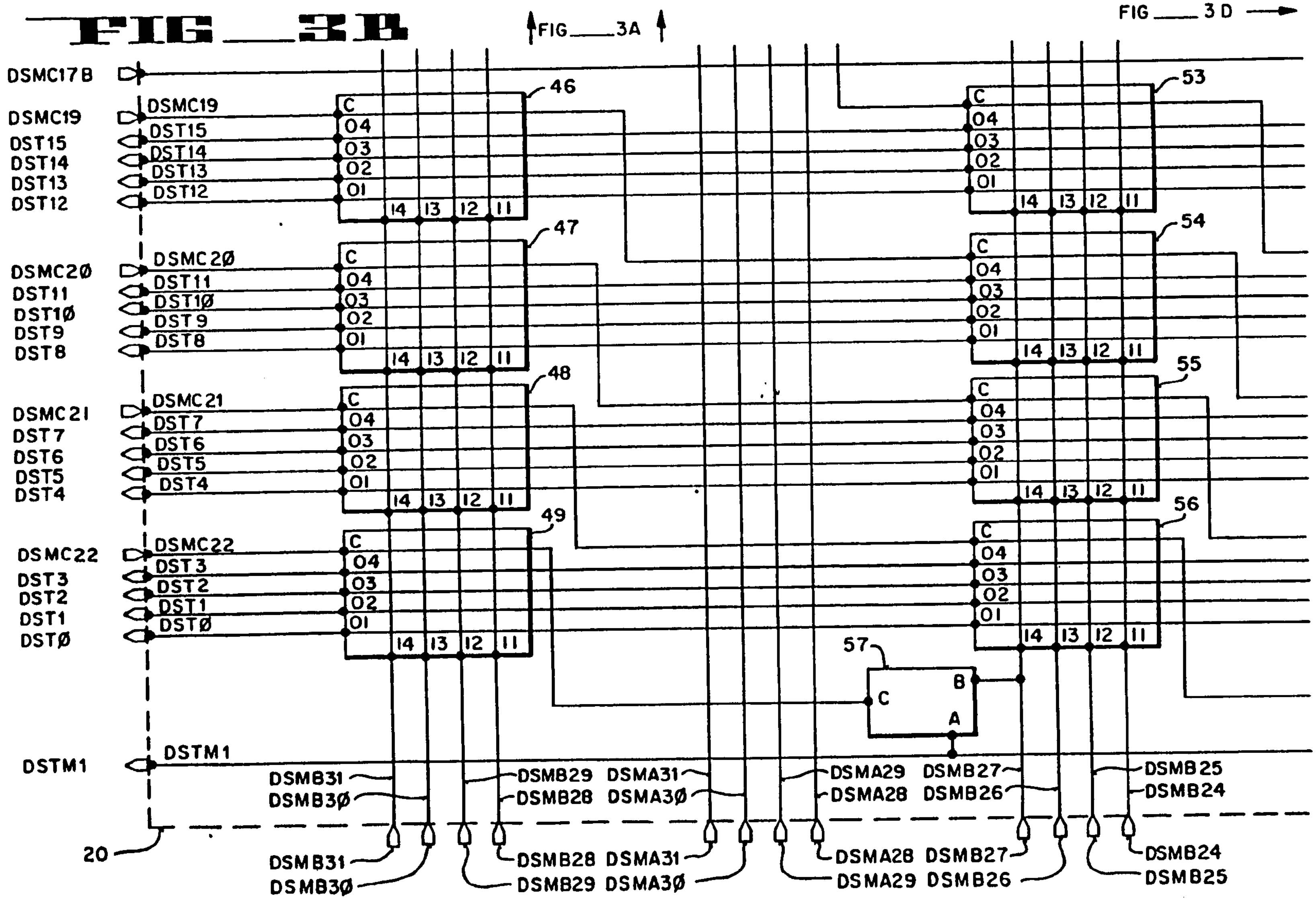


FIG 3A
FIG 3D
DSMC17B
DSMC19
DST15
DST14
DST13
DST12
DSMC2Ø
DST11
DST1Ø
DST9
DST8
DSMC21
DST7
DST6
DST5
DST4
DSMC22
DST3
DST2
DST1
DSTØ
DSTM1
46
47
48
49
53
54
55
56
57
C
O4
O3
O2
O1
I4
I3
I2
I1
B
A
DSMB31
DSMB3Ø
DSMB29
DSMB28
DSMA31
DSMA3Ø
DSMA29
DSMA28
DSMB27
DSMB26
DSMB25
DSMB24
20

FIG. 3C

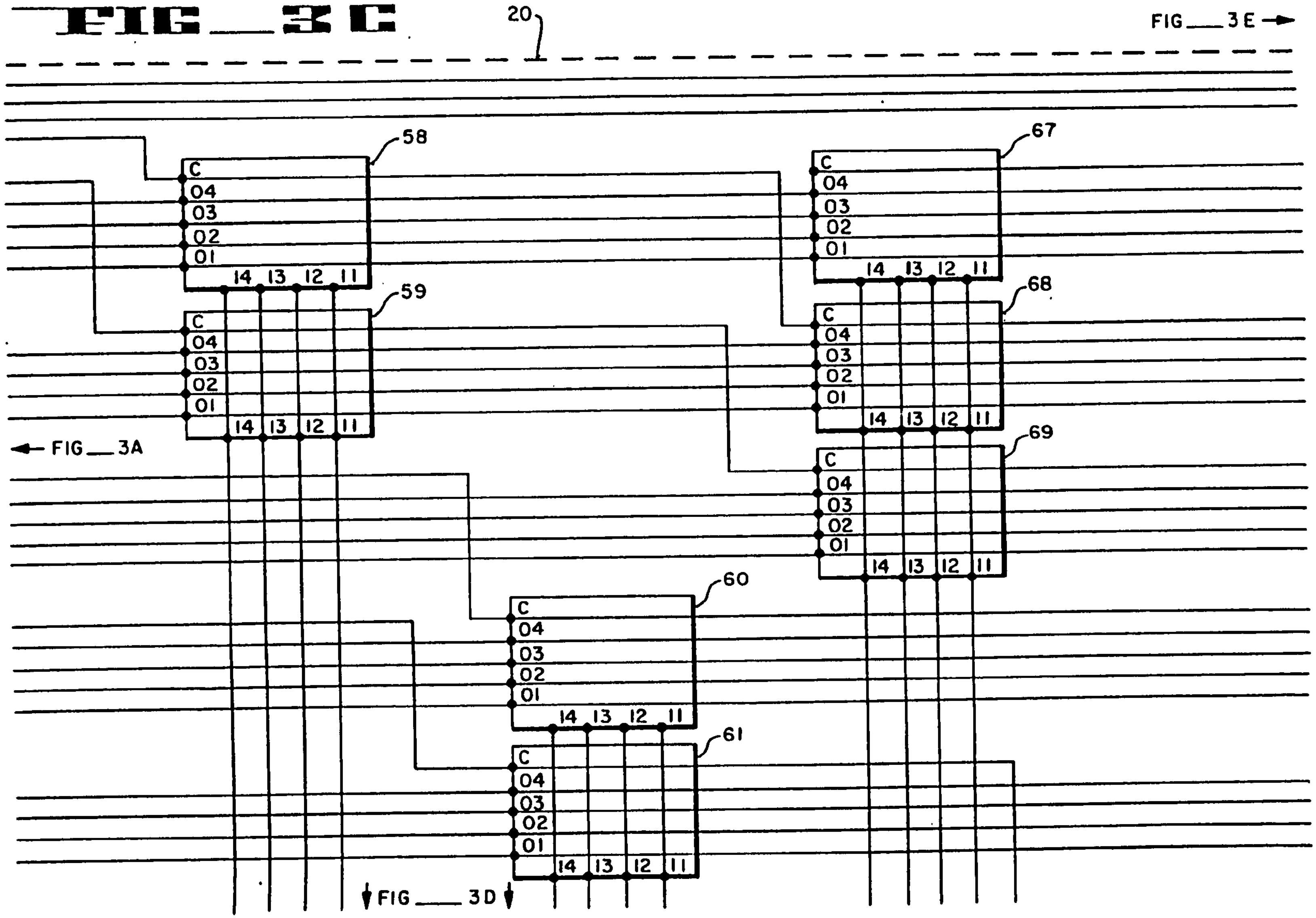


FIG 3D

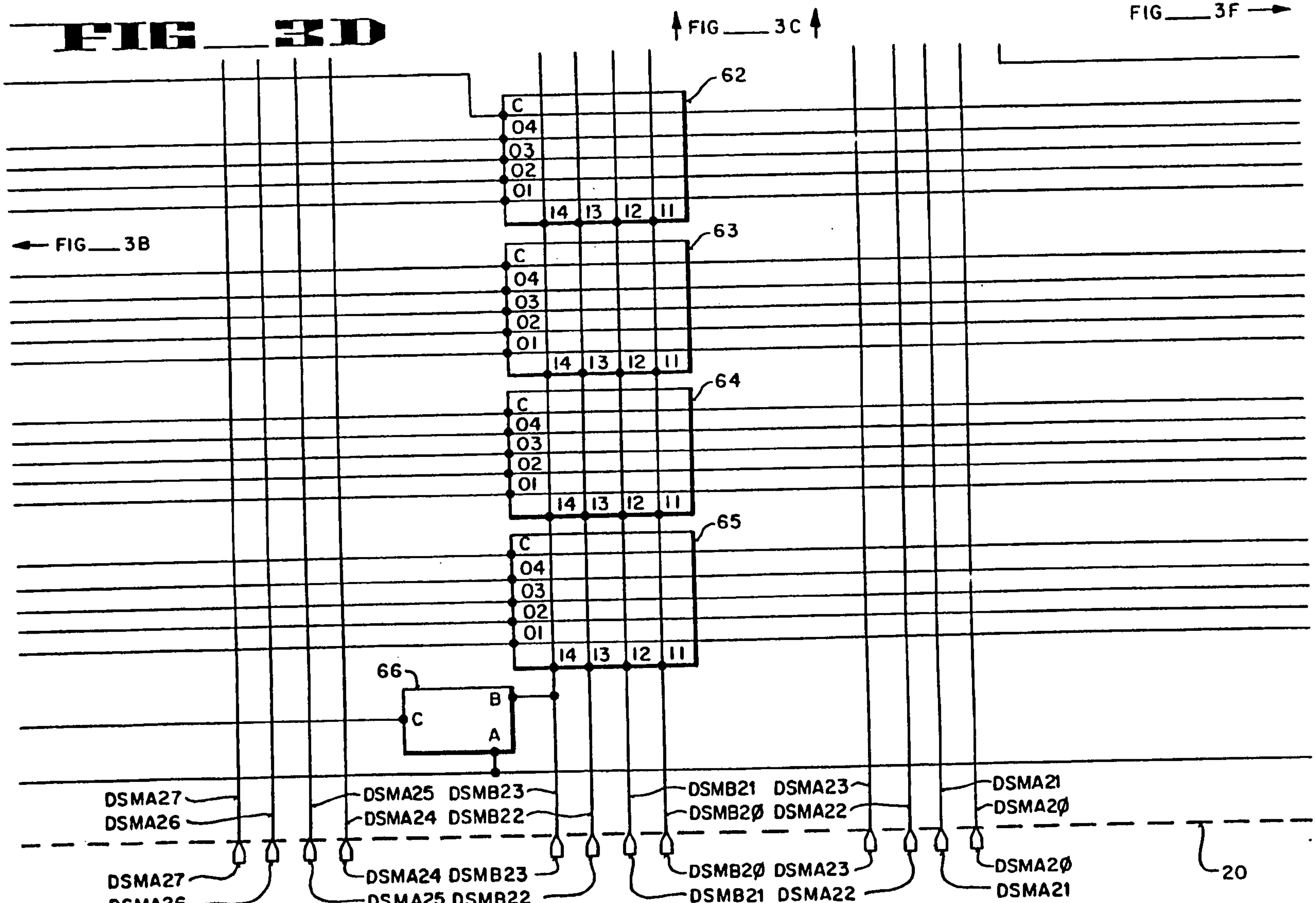


FIG 3C
FIG 3F
FIG 3B
62
63
64
65
66
C
O4
O3
O2
O1
I4
I3
I2
I1
B
A
C
DSMA27
DSMA26
DSMA25 DSMB23
DSMA24 DSMB22
DSMB21 DSMA23
DSMB2Ø DSMA22
DSMA21
DSMA2Ø
DSMA27
DSMA26
DSMA24 DSMB23
DSMA25 DSMB22
DSMB2Ø DSMA23
DSMB21 DSMA22
DSMA2Ø
DSMA21
20

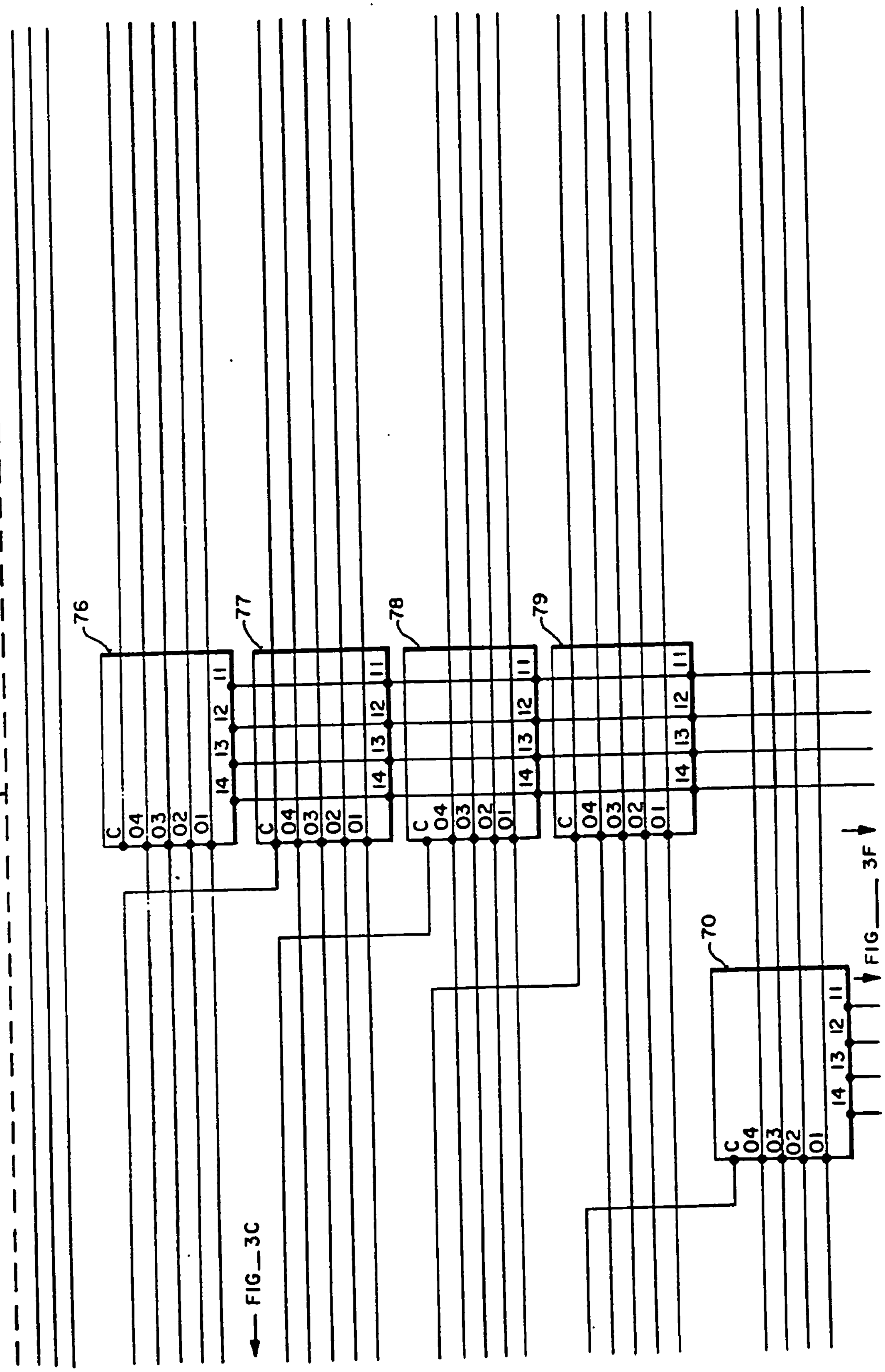


FIG 3E

FIG 3F

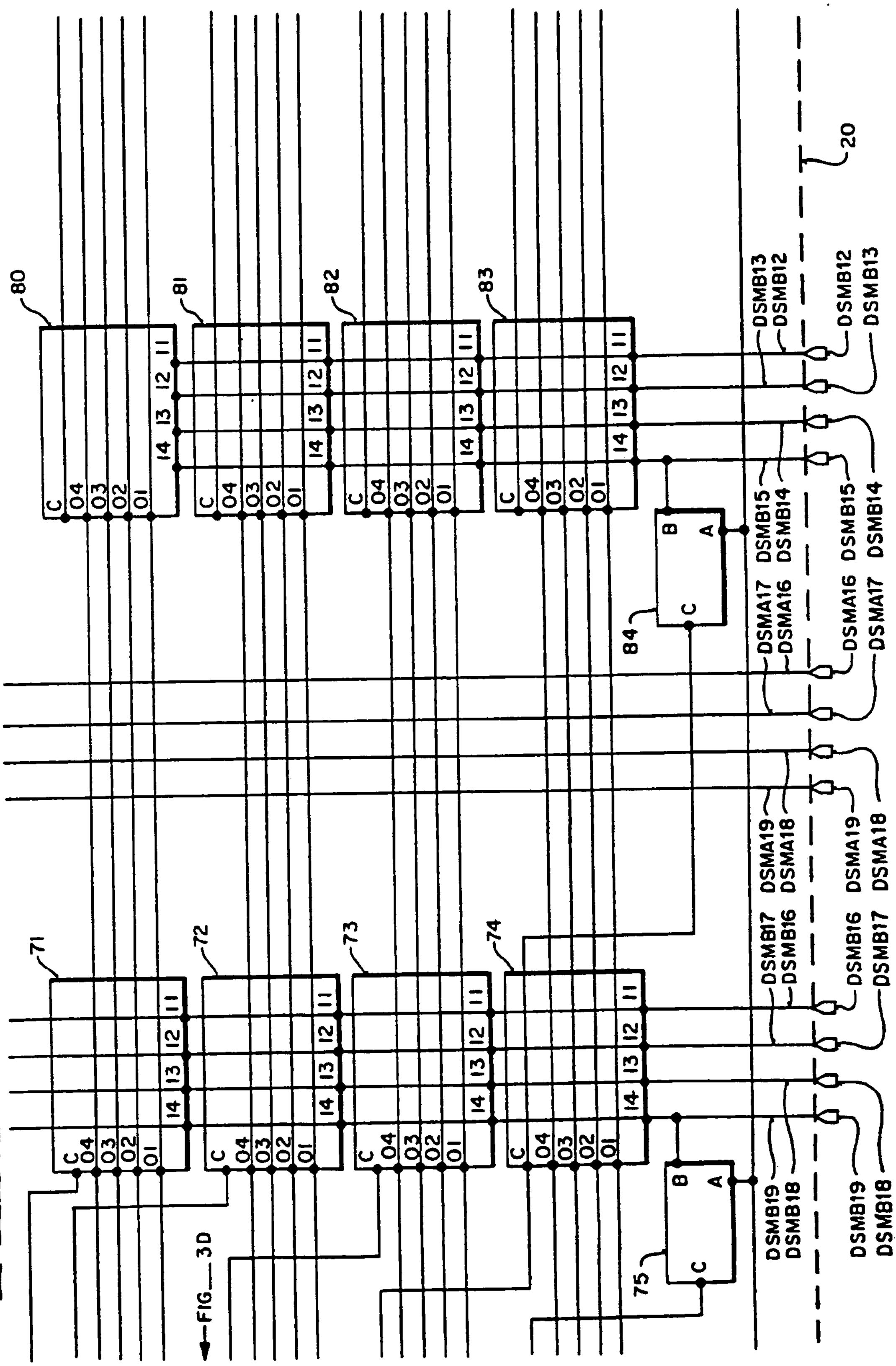


FIG 3H
FIG 3E
FIG 3D
71
72
73
74
75
80
81
82
83
84
20
C
O4
O3
O2
O1
I1
I2
I3
I4
A
B
DSMB12
DSMB13
DSMB14
DSMB15
DSMA16
DSMA17
DSMA18
DSMA19
DSMB16
DSMB17
DSMB18
DSMB19

FIG 3G

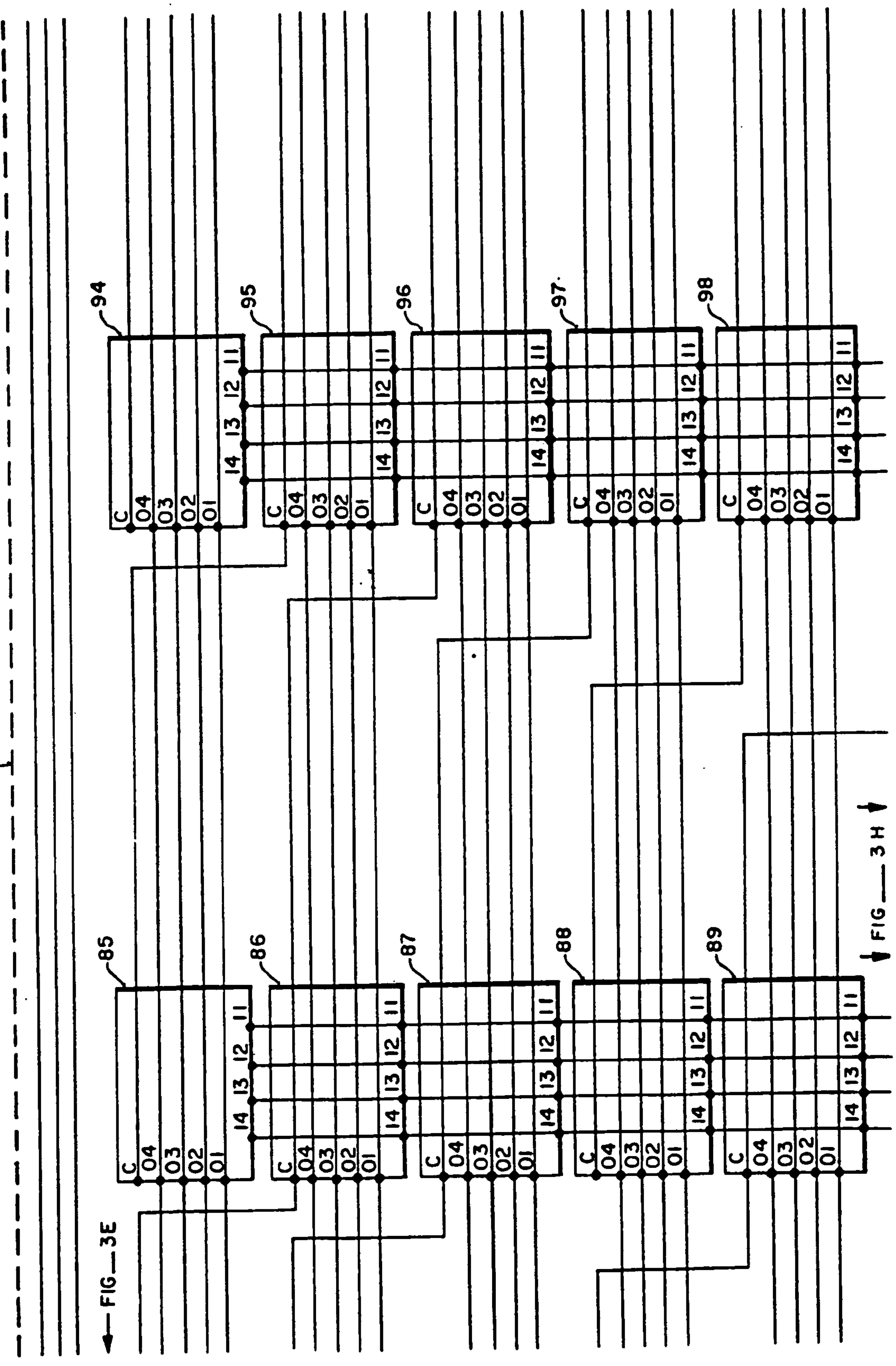


FIG 3E
FIG 3I
FIG 3H
20
85
86
87
88
89
94
95
96
97
98
C
O4
O3
O2
O1
I4
I3
I2
I1

FIG 3J →
FIG 3G
← FIG 3F

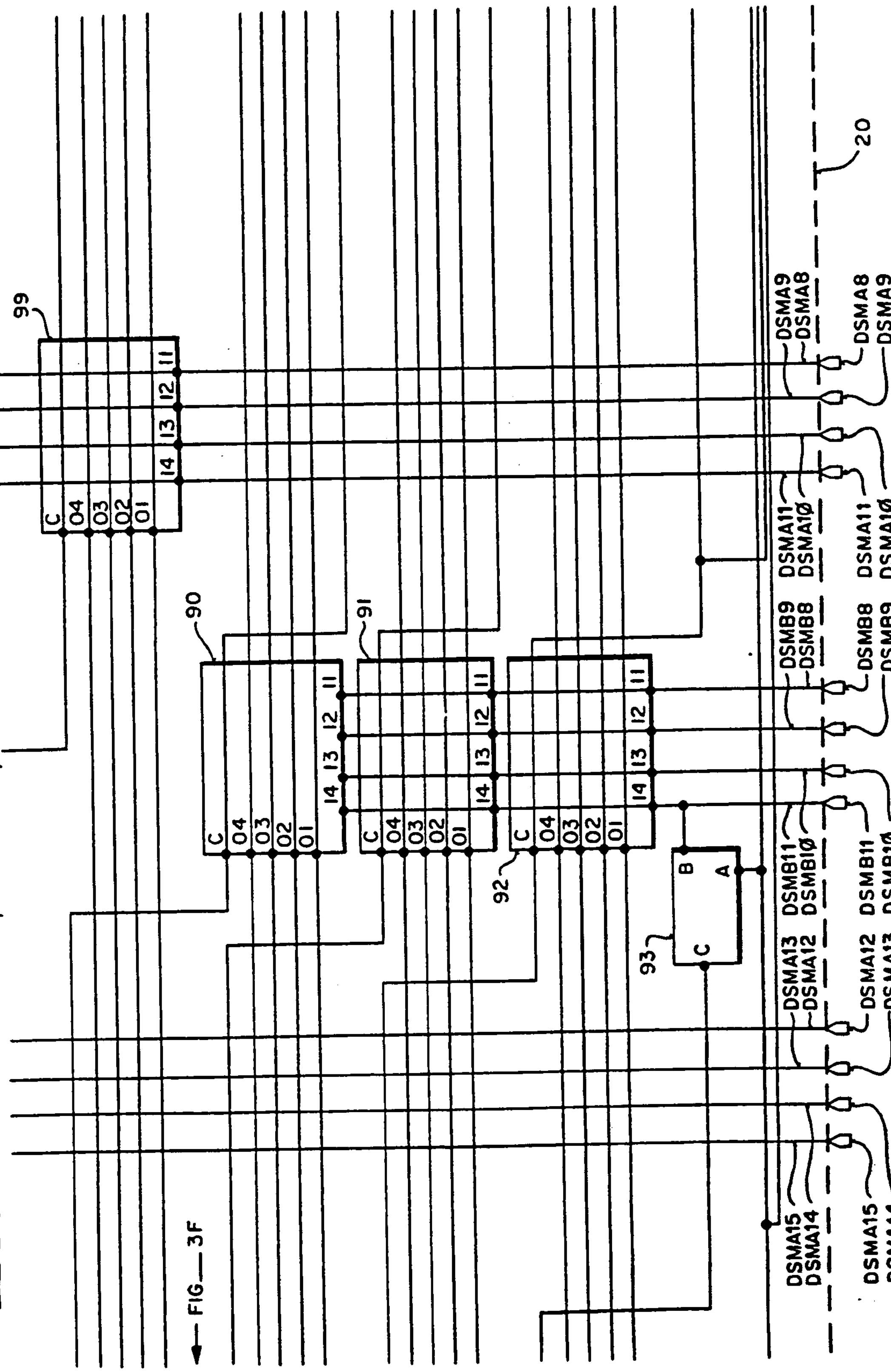


FIG 3H

FIG 3I

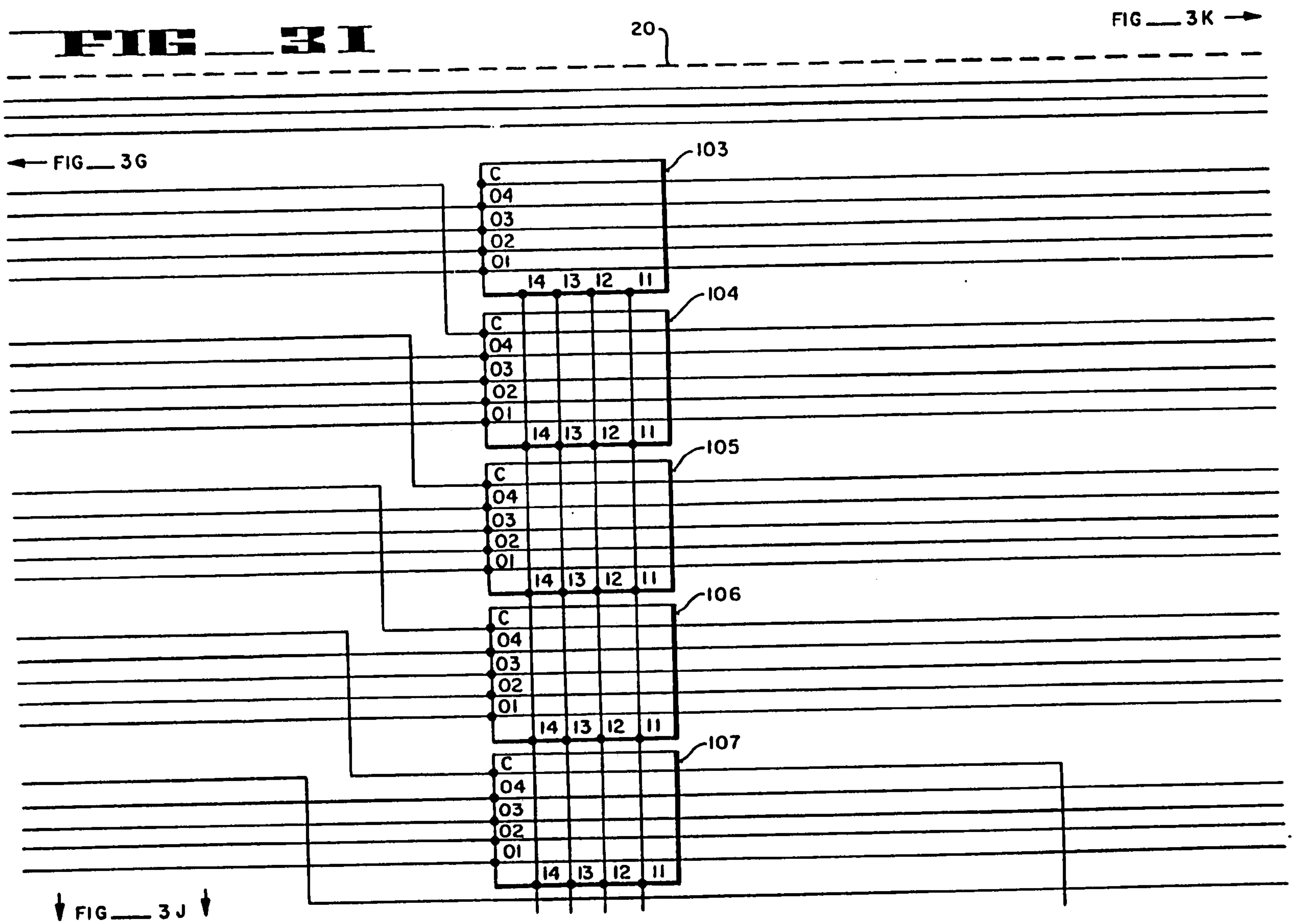


FIG. 3J

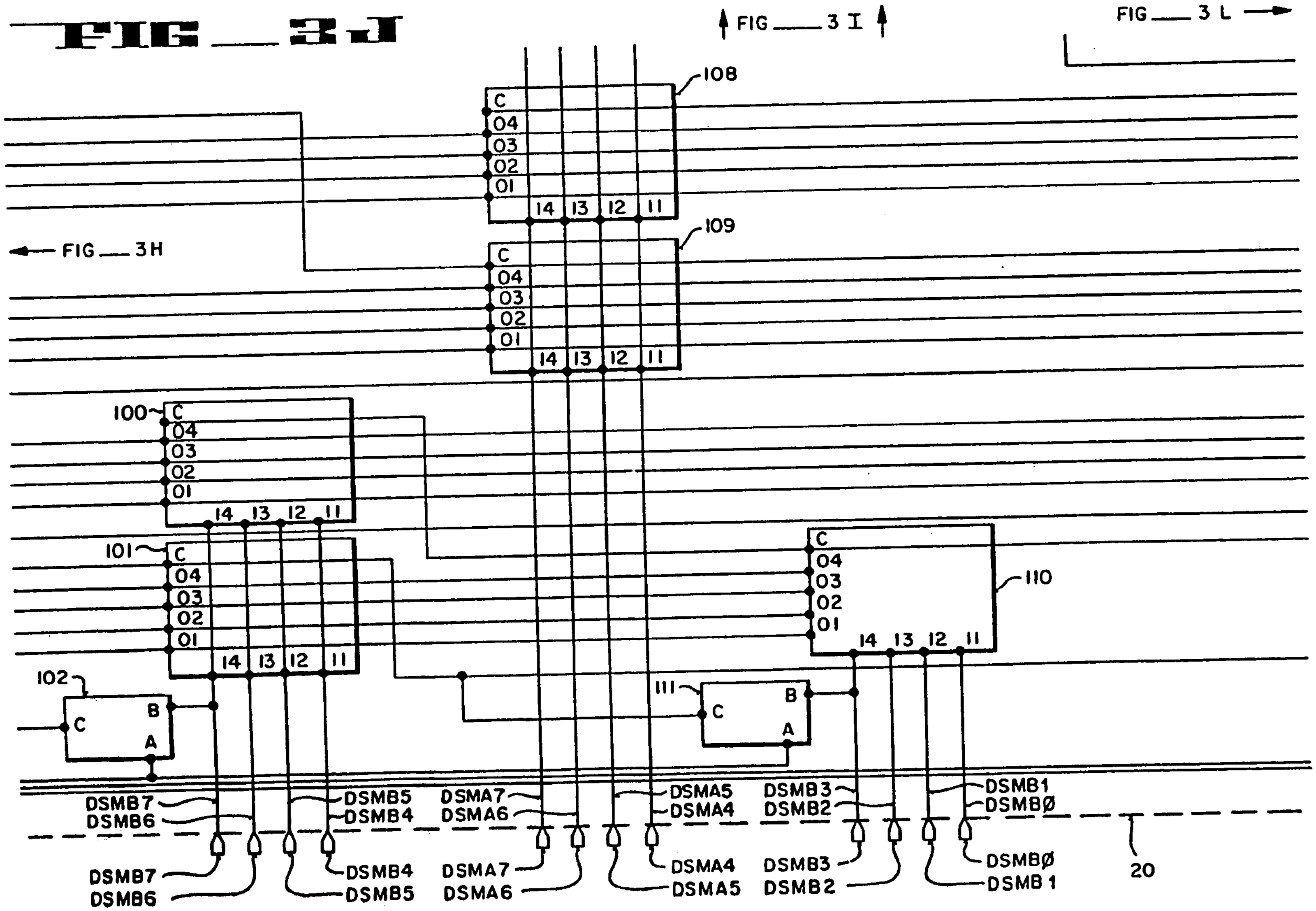


FIG. 3I
FIG. 3L
FIG. 3H
108
109
100
101
102
110
111
C
04
03
02
01
14
13
12
11
B
A
DSMB7
DSMB6
DSMB5
DSMB4
DSMA7
DSMA6
DSMA5
DSMA4
DSMB3
DSMB2
DSMB1
DSMBØ
20

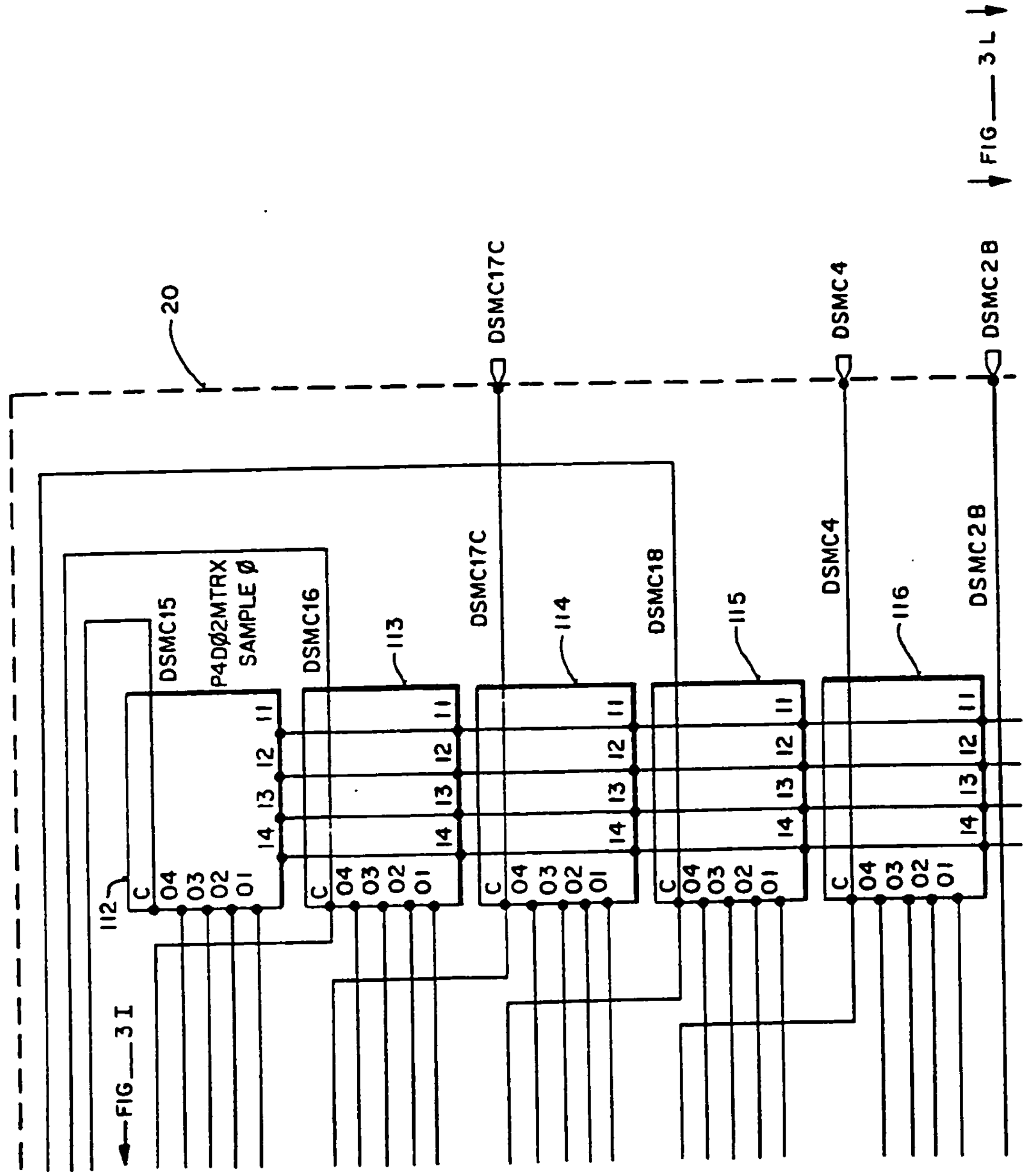


FIG 3L
20
DSMC17C
DSMC4
DSMC2B
DSMC15
P4DØ2MTRX
SAMPLE Ø
DSMC16
113
DSMC17C
114
DSMC18
115
DSMC4
116
DSMC2B
112
C
04
03
02
01
11
12
13
14
FIG 3I

FIG 3K

FIG 3L

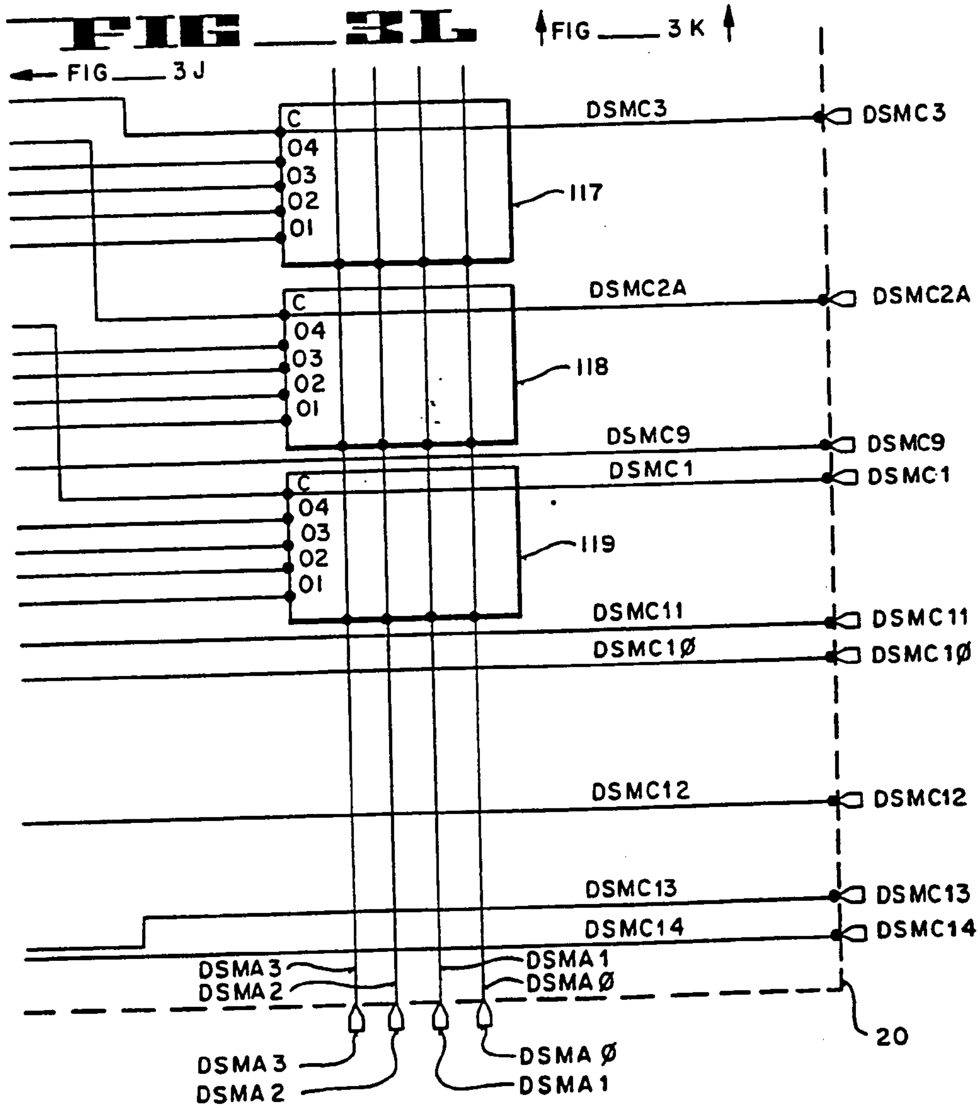
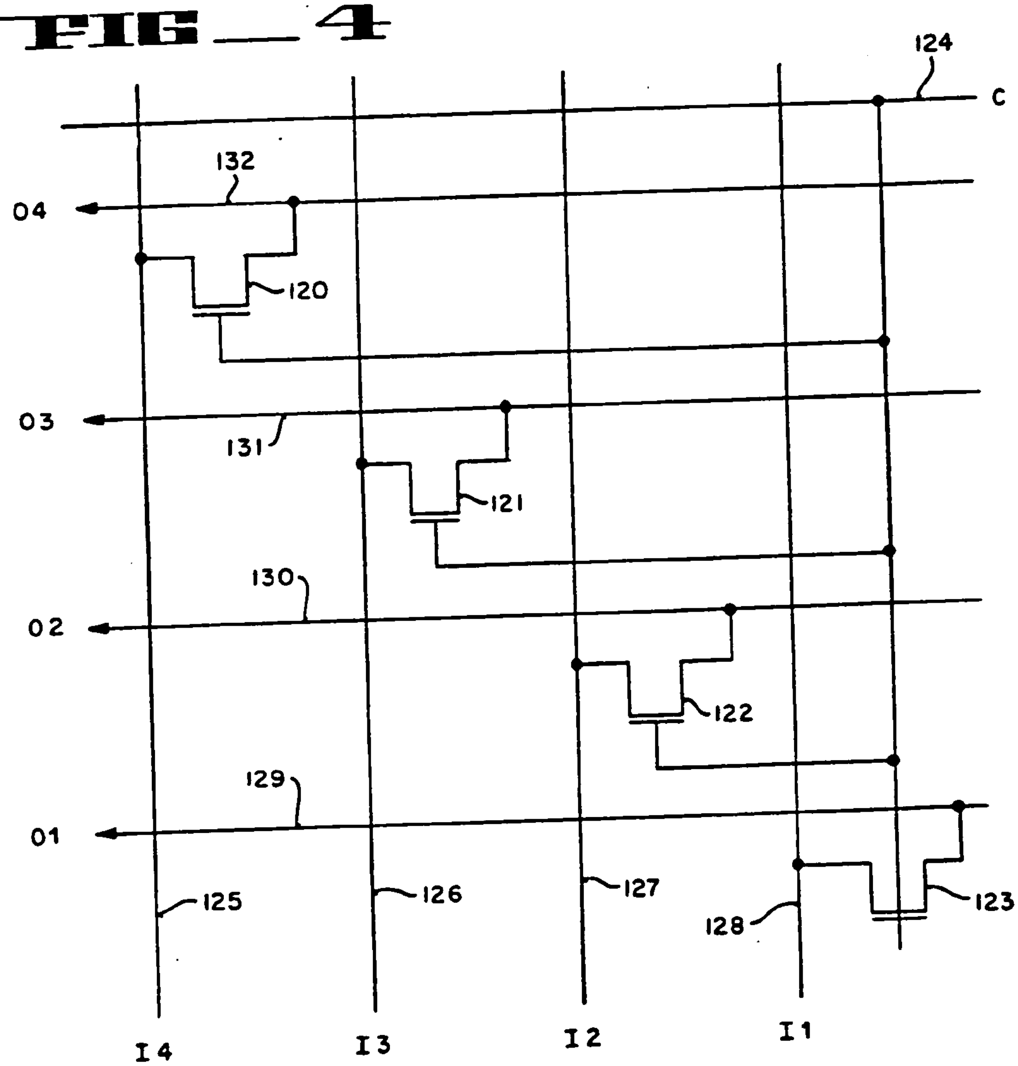


FIG 3K
FIG 3J
C
04
03
02
01
117
118
119
DSMC3
DSMC2A
DSMC9
DSMC1
DSMC11
DSMC1Ø
DSMC12
DSMC13
DSMC14
DSMA 3
DSMA 2
DSMA 1
DSMA Ø
20

FIG_4


124
C
132
O4
120
O3
131
121
130
O2
122
129
O1
125
126
127
128
123
I4
I3
I2
I1

BYTE SWAP INSTRUCTION FOR MEMORY FORMAT CONVERSION WITHIN A MICROPROCESSOR

FIELD OF THE INVENTION

The present invention relates to the field of semiconductor microprocessors.

BACKGROUND OF THE INVENTION

The present invention covers a byte swapping instruction which may be implemented within the architecture of a microprocessor. The microprocessor utilized with the present invention is the Intel 80486™ Microprocessor, frequently referred to as the 486™ Processor. The 486 processor is an improved version of the Intel 80386™ microprocessor, also known as the 386™ processor. (Intel, 80386, 386, 80486 and 486 are trademarks of Intel Corporation).

Generally, information is stored in the memory of a microprocessor system in data structures which typically vary anywhere between 8 to 64-bits in length. In the 486 microprocessor a "word" is defined to be 16-bits wide, while a doubleword, or "dword", is 32-bits wide. Words are stored in two consecutive 8-bit bytes in memory with the low-order byte at the lowest address and the higher-order byte at the higher address. Dwords are stored in four consecutive bytes in memory with the low-order byte at the lowest address and the high-order byte at a highest address. The address of a word or dword data item within the microprocessor is the byte address of the lowest-order byte. This type of addressing, particularly with respect to a dword data item, is known as the "little-endian" method for storing data types that are larger than one byte. All of Intel's x86 family members use the little-endian method for storing data types.

The alternative method of storing data types within a memory of a microprocessor is referred to as the "big-endian" method. In the big-endian

method, data is stored with the high-order bits at the lowest addressed byte. Thus, the big-endian format is opposite to the little-endian counterpart. The distinction between the two is simply which byte of a multiple byte quantity is assigned the lowest address, and which byte is assigned the highest address. In big-endian format, as the name implies, the big bytes come first; that is, the high-order bits are at lower addresses. The big-endian memory format is used by IBM's 370 line of computers as well as the 68000 line of microprocessors manufactured by Motorola, Inc. In addition, many RISC processors use the big-endian format.

Very often a programmer desires to form a data base having mixed data memory formats. Other programmers frequently want to send data over a network from one computer which stores integer data in a big-endian format to another computer which stores integer data in a little-endian format. Therefore, at some point in time, a conversion needs to be performed to convert data stored in one memory format to the other.

In a 16-bit environment the conversion between memory formats is straightforward. A number of instructions are generally available within a microprocessor to simply rotate or exchange 8-bit registers. In other words, the 8-bit quantities that form the 16-bit data item can simply be swapped or exchanged.

Byte swaps of higher-order number of bits, say 32 or 64-bit quantities, are more problematic. For example, one way that a prior art microprocessor might perform this byte swap operation on a 32-bit item is to first execute a byte swap of the lower two bytes; then rotate by sixteen; then perform a second bye swap on the remaining two bytes. Hence, three separate instructions are required to perform a conversion - - each instruction taking two clocks to implement for a total of six clocks for the entire conversion. Also, because each instruction is generally two to three bytes in length, a great deal of code needs to be generated -- probably nine instruction bytes -- for these three rotate instructions.

An alternative approach would be to have the memory format

conversation take place in consecutive steps in microcode. However, using microcode would still take six clocks or more along with a large number of instruction bytes. Consequently, performing memory format conversions from big-endian to little-endian, or visa-versa, in prior art machines requires a substantial amount of internal memory space and a significant performance penalty.

A different approach that is used by certain RISC processors is referred to as "pin-strapping". Pin-strapping consists of nothing more than a static switch that is hard-wired into the printed circuit board housing the microprocessor. The pin-strap option forces the computer to treat memory in one fashion or another, i.e., either as big-endian or little-endian format. This hard-wired approach has the obvious drawback in that it is static and therefore incapable of being programmed or controlled dynamically by the microprocessor or user.

As will be seen, the present invention replaces these past approaches with a single byte swap instruction capable of converting a big-endian dword to a little-endian format. This instruction provides rapid conversion between the two formats without adding any extra hardware or performance cost. An approximately 10% speed increase is reported for programs that make heavy use of big-endian data when executing on a 486 processor (e.g.; a little-endian machine).

SUMMARY OF THE INVENTION

A specialized microprocessor instruction optimized for performing an in-place byte swap on 32-bit data type is described. This byte swap operation is especially useful in converting data stored in a big-endian memory format to a little-endian memory format, or visa-versa. The invention comprises a modified barrel shifter which includes a plurality of multiplexers for selectively coupling data from one or more input buses to an output bus. The coupling of the individual bit lines of the data buses is arranged such that bits 0-7, 8-15, 16-23 and 24-31 of the input bus are coupled to corresponding bits 24-31, 16-23, 8-15 and 0-7, respectively. Control lines connected to each of the multiplexers provide a means for controlling the byte swapping operation.

The presently described byte swap instruction allows the programmer to convert data from a big-endian memory format to a little-endian data format, and back again, without incurring the performance penalties associated with past microprocessors. In addition, this one instruction requires only one execution clock cycle to perform the conversion whereas prior art microprocessors typically require three instructions and six clocks .

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of the preferred embodiment of the invention which, however should not be taken to limit the invention to the specific embodiment but are for explanation and understanding only.

Figure 1 is a comparison between little-endian and big-endian memory formats. Both memory formats are shown with their corresponding memory addresses and bit numbering. The highest order bit is shown as bit 31 while the lowest order bit is bit 0.

Figure 2 is a block diagram illustrating the data flow through the barrel shifter of the invented microprocessor.

Figures 3A - 3L collectively comprise a circuit schematic of the barrel shifter utilized in the currently preferred embodiment of the present invention.

Figure 4 shows the circuit schematic for the 4 x 4 multiplexer utilized within the barrel shifter of Figure 3.

DETAILED DESCRIPTION OF THE INVENTION

A microprocessor comprising a byte swap instruction for converting the memory data formats from one type to another is described. In the following description, numerous specific details are set forth, such as bit lengths, etc., in order to provide a thorough understanding of the present invention. It will be obvious however, to one skilled in the art that these specific details need not be used to practice the present invention. In other instances, well-known structures and circuits have not been shown in detail in order not to unnecessarily obscure the present invention.

Figure 1 illustrates the differences between the big-endian and little-endian memory formats for dwords having a length of 32-bits. In Figure 1, for both the little-endian and big-endian memory formats, the 32-bits of data are shown with the low-order bit numbered bit 0, the high-order bit numbered bit 31, and the memory addresses numbered along the top.

As shown, each 32-bit dword is partitioned into four 8-bit bytes. These are denoted by capital letters A-D in Figure 1. In the little-endian memory format, dwords are stored in four consecutive bytes in memory with the low-order byte being positioned at the lowest address and the high-order byte positioned at the highest address. This is illustrated in Figure 1 where for a little-endian memory format, bits 0-7 are stored in memory address M, bits 8-15 are stored in memory address M+1, bits 16-22 are stored in memory address M+2, and bits 24-31 are stored in memory address M+3. The address of a dword data item in a little-endian format is the byte address of the lowest-order byte, (e.g., memory address "M").

In the big-endian memory data format, the bits are arranged in the opposite order. That is, bit-endian data is stored with the high-order bits at the lowest address byte, and the lowest-order bits at the highest memory address byte. Therefore, as shown in Figure 1, in big-endian format bits 0-7 ar stored at memory address M+3; bits 8-15 are stored at memory

address M+2; bits 16-23 are stored at memory address M+1; and bits 24-31 are stored at memory address M. The address of a dword data item in big-endian memory format is the byte address of the highest-order byte.

To perform a conversion from big-endian memory format to little-endian, the following process needs to occur. First, the data item is moved from memory to an internal register. Next, byte A corresponding to bits 24-31 in the big-endian format need to be transferred or swapped with the contents of byte D corresponding to bits 0-7. Similarly, byte B needs to be swapped with byte C. This byte swapping operation is performed in the preferred embodiment using the barrel shifter located within the integer execution unit of the microprocessor. Finally, the swapped data item is latched into a temporary latch or register to be subsequently written back to the source destination register in memory.

Referring now to Figure 2, a block diagram of the data flow through the barrel shifter utilized to implement the byte swap instruction of the present invention is shown. The barrel shifter 20 is a device commonly known in the art, and is typical found in most microprocessors. Barrel shifter 20 can shift/rotate a word of data by *N* positions in a single operation -- *N* ranging from 0 to the word size. Ordinarily, barrel shifter 20 is used for a variety of operations. For instance, barrel shifter 20 is used for shift instructions, rotate instructions, bit scans, etc.

In the invented microprocessor, barrel shifter 20 is comprised of two parts: a matrix element 23 and a tree 24. Matrix 23 receives data inputs along 32-bit buses 21 and 22, labelled *dsma* and *dsmb*, respectively. Generally, the data supplied along bus line 21 is identical to the data supplied on bus line 22. This facilitates rapid rotation or shifting of data as will be seen. Matrix 23 receives these data inputs and performs a shift operation on a nibble granular basis, (i.e., the data is shifted by a multiple of 4). In this way, the data is initially shifted in the first stage of the shift operation by multiples of 4 (e.g., 4, 8, 12, 16, etc.). Additional shifting by anything less than 4 (i.e., 0-3) is performed in tree 24. Matrix 23 is coupled

to tree 24 along bus 25 labelled DST. Tree 24 then provides an output on 32-bit bus 26 labelled DSL to ALU output register 27.

Referring now to Figure 3A - 3L, a detailed circuit schematic of the barrel shifter 20 utilized in the currently preferred embodiment of the present invention is illustrated. Barrel shifter 20 comprises a plurality of vertical data input lines coupled to *dsma* and *dsmb* bus lines 21 and 22, respectively. These data input lines are shown comprised of individual bit lines *DSMA0-31* and *DSMB0-31*. Each line is coupled to the inputs of a plurality of 4x4 multiplexers. For example, *dsmb31* is coupled to the I4 input of multiplexors 42-49.

Also included in Figures 3A-L are a plurality of horizontal data output lines. These horizontal data bit lines comprise DST bus 25 and are coupled to the outputs of a plurality of multiplexers. The individual bit lines of DST bus 25 are shown labelled *DST0-31* and are connected to ALU output register 27 as discussed in connection with Figure 2 (additional lines are shown, e.g., *DST35*, but these are not germane to the invention; therefore, they will not be discussed). For instance, *DST31* is shown connected to multiplexers 42, 59, 68, 77, 86, 95, 104 and 113. Control is provided to each of the multiplexers of barrel shifter 20 via control lines labelled *DSMC0-22*..

Each of the multiplexers 41-119 (excluding devices 57, 66, 75, 84, 93, 102 and 111 which are only used in conjunction with tree 24) comprise a multiplexer having four input pins (i.e., I1-I4), four output pins (i.e., O1-O4), and a control pin (i.e.,C). During operation, when the control line of an individual multiplexer is asserted, the input data lines are electrically coupled to the output data lines. By way of example, when the control input line to multiplexer 110 is asserted the data present at input pin I_1 (e.g., *DSMB0*) is electrically coupled to output pin O_1 , (e.g., *DST0*). In a similar manner, the data associated with pins I_2, I_3, and I_4 (e.g., *DSMB1, DSMB2,* and *DSMB3*, respectively) is electrically coupled to output pins O_2, O_3 and O_4, respectively (e.g., *DST1*, *DST2*, and *DST3*, respectively). Thus, each

individual multiplexer within barrel shifter 20 operates as a switching element connecting groups of data inputs to corresponding output lines.

With reference now to Figure 4, a circuit schematic of the 4 x 4 multiplexer used in the currently preferred embodiment of the present invention is shown. This multiplexer includes field-effect devices 120-123 which preferably comprise ordinary n-channel MOS devices. The gate of each of these field-effect transistors is coupled to control line 124. The drain and source regions of each transistor are coupled to a different pair of input and output lines. For instance, transistor 123 has its source connected to input line 128 labelled I_1 and its drain coupled to output line 129 labelled 0_1. When control line 124 is asserted by taking it to a high positive potential, a conductive channel is formed between the source and drain regions of devices 120-123. This conductive channel provides electrical connection between the corresponding input and output pins. Thus, a high positive potential on control line 124 provides electrical connection between lines 128 and 129, 127 and 130, 126 and 131, and 125 and 132.

Barrel shifter 20 of the invented microprocessor is similar to barrel shifters found in prior art microprocessors such as the 80386 with the exception that certain control signals have been split to accommodate the format conversion of the present invention. The original matrix control signal *DSMC2* of the 386 processor has been split into three control signals, *DSMC2a*, *DSMC2b* and *DSMC2c* in order to aid the one execution clock implementation of the byte swap (BSWAP) instruction. Also, the *DSMC17* signal is shown split into three signals, *DSMC17a-c*. For ordinary operations (everything except a BSWAP) these control lines are merged into their original form and barrel shifter 20 operates normally. It is only during execution of the BSWAP instruction that the separate split lines become important.

In phase 1 of the execution clock cycle of the BSWAP instruction, four of the matrix control signals are asserted. All other control signals of the

matrix are negated. The four matrix control signals which are asserted include *dsmc21* (which muxes bits 24-31 of *dsmb* bus to bits 0-7 of the matrix output bus DST), *dsmc17b* (which muxes bits 16-23 of *dsmb* bus to bits 8-15 of matrix output bus DST), *dsmc2b* (muxing bits 8-15 of *dsma* bus to bits 16-23 of matrix output bus DST) and *dsmc17c* (muxing bits 0-7 of *dsma* bus onto bits 24-31 of matrix output bus DST). In phase 2 of the execution clock, the barrel shifter accepts the data inputs along the *dsma* and *dsmb* buses and swaps the bytes simply by providing appropriate electrical connection between the *DSMA*, *DSMB* buses and *DST* output lines. Therefore, the memory item data format is automatically converted from either big-endian to little-endian or from little-endian to big-endian. The converted data output by the matrix proceeds through the tree of the shifter without any additional shifts and is subsequently loaded into the ALU output latch 27.

In the currently preferred embodiment, the BSWAP instruction is comprised by two 8-bit instruction bytes. Thirteen of the bits of the instruction are defined as the opcode while the lower four bits of the second byte specify the register that is to participate in the operation.

One important aspect of the present invention is that the BSWAP instruction acts as its own inverse so that a separate instruction to convert back from a previously converted format is unnecessary. In other words, the data provided on data input buses 21 and 22, may be converted from big to little, or from little to big, using the same instruction.

Another operation where the BSWAP instruction proves useful is in load/store operations. Consider the case in which a user wishes to perform operations on a machine which stores all data in a little-endian format. If the user performs a normal load to a register followed by a BSWAP instruction, the big-endian format is translated into little-endian automatically, thereby enabling the processor to operate on it immediately (i.e., add, subtract, multiply, etc.). Following completion of all arithmetic operations, the programmer might execute another BSWAP followed by a normal store to

memory to restore the data in its original format in memory. Achieving the identical sequence of operations on a prior art microprocessor such as the 80386 takes considerably more time and storage area.

Although the byte swap instruction of the present invention is currently defined for 32-bit dwords, it also provides a basic building block for higher-order memory format conversions (e.g., 64-bits and on). In executing a 64-bit swap, the machine first loads the 64-bit quantity into two registers. Next, it executes BSWAPs on each of the registers. Afterwards, the processor simply renames the registers in reverse order. In other words, the register containing the higher-order bits is renamed as the lower-order register, and visa-versa. This type of renaming scheme obviates the need to physically swap the registers. When utilized for conversions of 64-bits and higher, the performance cost savings are magnified.

Thus, a byte swap instruction in a microprocessor for converting between memory data formats has been described.

CLAIMS

1. An instruction for swapping the byte order of a 32-bit data item stored in a register of a microprocessor, said instruction comprising:

coupling means for coupling said data item on a plurality of input data lines to a corresponding plurality of output data lines, the coupling arranged in a manner such that bits 0-7, 8-15, 16-23 and 24-31 of said data item are coupled to corresponding bit positions 24-31, 16-23, 8-15 and 0-7, respectively, of said output data line; and

control means coupled to said coupling means for controlling said swapping of said byte order of said data item.

2. The instruction of Claim 1, further comprising a register means coupled to said output data lines for storing said swapped data item.

3. The instruction of Claim 2, wherein said coupling means comprises a barrel shifter which includes a plurality of multiplexers each coupling at least one bit of said data item on said input data lines to said corresponding output data line.

4. The instruction according to Claim 2, wherein said instruction comprises two instruction bytes.

5. The instruction according to Claim 2, wherein said swapping of said byte order is performed in one execution clock cycle.

6. A microprocessor having a memory, a barrel shifter for shifting a dword of data, said shifter receiving data inputs along first and second buses and producing a shifted or rotated output along a third bus, and an instruction for changing the format of a 32-bit data item from a first format to a

second format, said instruction comprising:

a barrel shifter including a plurality of input lines, a plurality of output lines and a plurality of multiplexers, said multiplexers electrically connecting ordered bits 0-7, 8-15, 16-23, and 24-31 to corresponding ordered bit positions 24-31, 16-23, 8-15, 0-7, respectively, of said output data lines when said multiplexers are enabled; and

control means coupled to said plurality of multiplexers for enabling said multiplexers.

7. The microprocessor of Claim 6 in which said instruction also changes the format of said data item from said second format to said first format.

8. The microprocessor of Claim 7 wherein said first format is big-endian and said second format is little-endian.

9. The microprocessor of Claim 8, wherein said instruction comprises two instruction bytes.

10. The microprocessor of Claim 7 wherein said changing of said formats is performed in one execution clock cycle.

11. In a microprocessor having an internal register set and a barrel shifter, a method of converting a data item stored in a first memory format to a second memory format comprising the steps of:

inputting said data item into said barrel shifter along a pair of input buses;

asserting certain control signals connected to said barrel shifter to initiate a conversion;

connecting the ordered bits 0-7, 8-15, 16-23 and 24-31 of said data item along said input buses to the corresponding ordered bit positions

24-31, 16-23, 8-15 and 0-7, respectively, of said output data bus, thereby producing a converted data item.

12. The method of Claim 11 further including the step of latching said converted data item into a storage location.

13. In a computer having a memory in which data is stored in a first memory format, a method of converting a 32-bit data item to a second memory format comprising the steps of:

moving said data item from said memory to a register;

swapping the byte order of said data item such that ordered bits 0-7, 8-15, 16-23 and 24-31 of said data item in said first format correspond to the ordered bit positions 24-31, 16-23, 8-15 and 0-7, respectively, of said register, said swapped data item being in said second format; and

latching said swapped data item into a latch.

14. The method of Claim 13 wherein said computer further comprises a barrel shifter having a plurality of input bit lines and a plurality of output bit lines, and wherein said swapping step comprises the steps of:

asserting control signals coupled to said barrel shifter to electrically connect the ordered bit positions 0-7, 8-15, 16-23 and 24-31 of said input bit lines to the corresponding ordered bit positions 24-31, 16-23, 8-15 and 0-7, respectively, of said output bit lines; and

reading said data item onto said input bit lines.

15. The method of Claim 14 wherein said swapping step occurs in one execution clock cycle.

16. The method of Claim 15 wherein said first format is little-endian and said second format is big-endian.

17. An instruction for swapping the byte order of a 32-bit data item stored in a register substantially as hereinbefore described with reference to the accompanying drawings.

Published 1990 at The Patent Office. State House 66/71 High Holborn, London WC1R 4TP. Further copies may be obtained from The Patent Office Sales Branch, St Mary Cray, Orpington, Kent BR5 3RD. Printed by Multiplex techniques ltd, St Mary Cray, Kent. Con. 1/87